CprE 288 – Introduction to Embedded Systems
(UART Interface Overview)

Instructors:
Dr. Phillip Jones
Announcement

• HW 5: Not graded
  – **CPRE 288: Datasheet Trainer**
• Lab 5: UART
• Exam 1: In class October 5 (Thursday, 10/5).
• Formal in class activity next Tuesday (9/26)
Overview of the Lecture

• Concepts behind Serial Communication
• TM4C123g UART Programming Interface
  – Textbook reading:
    • Section 8.5
• Initializing UART, transmitting and receiving data
Serial Communication

- UART = Universal Synchronous & Asynchronous Serial Receiver & Transmitter
- Asynchronous (no common clock)
- Can transmit over long link distances
- Uses start and stop to sandwich data bits
- parity bit can be used for error detection

- (on right) RS-232 Serial Cable

http://class.ece.iastate.edu/cpre288
## Serial Byte Format

### Logic 1

- Sampled here

### Logic 0

- Sampled here

<table>
<thead>
<tr>
<th>State</th>
<th>Idle</th>
<th>Idle</th>
<th>Active</th>
<th>Active</th>
<th>Active</th>
<th>Active</th>
<th>Active</th>
<th>Active</th>
<th>Active</th>
<th>Active</th>
<th>Active</th>
<th>Active</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Level</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Bit Type</td>
<td>Start</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Stop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Bit Number</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

Example: Sending byte value 01000001
Serial Communication

[Diagram showing a binary sequence with labels for start, stop, LSB (Least Significant Bit), MSB (Most Significant Bit), and voltage levels (+15V, +3V, -3V, -15V).]
Idle period: logic high
Start bit: logic low, 1 bit
Stop bit: logic high, 1 bit or 2 bits
Three choices: even, odd, or none

\[ P_{\text{even}} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0 \]

\[ P_{\text{odd}} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1 \]

*If one bit is flipped, how to detect it?*
How to define communication speed?

**Baud rate**: Number of symbols transferred per second
- Same as bit rate (bps) for USART

Baud rate is **not** data rate

With 56,000 bps, 8-data bit frame, two stop bits and parity bit bit used, what is the maximum data rate?
Diagram of UART Module (pg 644 in book)
Both sides of communication should use the same frame format and baud rate

Frame format:

– Number of **data** bits in the frame: 5, 6, 7, 8 or 9
– Number of stop bits: 1 or 2
– Parity bit: Odd, Even, or None
USART Programming Interface

Control and Status Registers:
UARTCTL, UARTCC, UARTLCRH, UARTFR
- 32-bit registers for control and status checking
- $n$ is 0 to 7, e.g. is UART0_CTL_R for USART0
- There are eight USART units; UART4 used for communication with iRobot Create and UART1 will be used in lab

Baud Rate Registers:
UARTIBRD, UARTFBRD
- Two 32-bit registers used together to set baud rate

32-bit Register for reading and writing data:
UARTDR
• Trying to set Control Registers?
  – Review pages 643 and 651 to 658 of the book

• Setting the baud rate registers?
  – See section 8.5.3.7 on page 650 of the book

• Need code examples for reading / writing data?
  – Figure 5.73 on Page 662 (explained on page 663) of the book
  – Also reproduced on upcoming slides
Serial (Tiva Launchpad)

- This time will be spent reviewing the individual bit positions inside of UARTCTL, UARTCC, UARTLCRH, UARTFR from pages 643 and 651 to 658 of the book
15 CTSEN – Enable clear to send
14 RTSEN – Enable request to send
11 RTS – Request to send
9 RXE – UART receive enable
8 TXE – UART transmit enable
7 LBE – UART loop back enable
5 HSE – High-Speed enable
4 EOT – End of transmission
3 SMART – Smart card support
2 SIRLP – UART SIR low-power mode
1 SIREN – UART SIR enable
0 UARTEN – UART enable
31:16, 13:12, 10, 6 Reserved – Read only
3:0 **UART** - baud clock source

31:4 Reserved
UARTLCRH: Control Register

7 SPS – UART stick parity select
6:5 WLEN – UART word length
4 FEN – UART enable FIFOs
3 STP2 – UART two stop bits select
2 EPS – UART even parity select
1 PEN – UART parity enable
0 BRK – UART send break
31:8 Reserved

Everything else will default to 1 stop bit, no parity, no FIFO
UARTFR: Status Register

7 TXFE – UART transmit FIFO empty
6 RXFF – UART receive FIFO full
5 TXFF – UART transmit FIFO full
4 RXFE – UART receive FIFO empty
3 BUSY – UART busy
0 CTS – Clear to send
31:8, 2:1 Reserved
Serial (TM4C123g)

- Baud rate
  - 1 baud = 1 symbol per second
  - In our case, 8 data bits are book ended by start and stop bits

- **Baud rate** is different from **data rate**
  - Baud rate includes overhead of start/stop/parity bits
Calculating Baud

• Two 32 bit registers UARTIBRD and UARTFBRD
• BRDI = integer portion, BRDF = fractional portion
• Baud Rate = $\frac{UARTSysClk}{(BRD) \times \text{ClkDiv}}$
  – $UARTSysClk = 16\text{Mhz}$
  – $\text{ClkDiv} = 16$ with HSE bit = 0 (8 with HSE bit = 1)
  – Baud Rate used in lab = 115200
• BRDI = (int)(BRD)
• BRDF = (int)(fraction of BRD)*64+.5
Example BRDI and BRDF

- Set a baud rate of 9600 bps for 16Mhz SysClk, HSE = 0
- BRD = 16,000,000 / (16 * 9600) = 104.16666
- BRDI = 104
- BRDF = .1666*64+.5 = 11.16666 = 11
/\texttt{Initialization part 1: GPIO (mostly)}\\

//Initialize USART1 to a given baud rate\\
\textbf{void uart_init(void)} {\\
  //enable clock to GPIO, R1 = port B\\
  SYSCTL_RCGCGPIO_R |\texttt{=} SYSCTL_RCGCGPIO\_R1;\\

  //enable clock to UART1, R1 = UART1. ***Must be done before setting Rx and Tx (See DataSheet)\\
  SYSCTL_RCGCUART_R |\texttt{=} SYSCTL_RCGCUART\_R1;\\

  //enable alternate functions on port b pins 0 and 1\\
  GPIO\_PORTB\_AFSEL\_R |\texttt{=} \texttt{(BIT0 | BIT1);}\\

  //enable Rx and Tx on port B on pins 0 and 1\\
  GPIO\_PORTB\_PCTL\_R |\texttt{=} 0x00000011;\\

  //set pin 0 and 1 to digital\\
  GPIO\_PORTB\_DEN\_R |\texttt{=} \texttt{(BIT0 | BIT1);}\\

  //set pin 0 to Rx or input\\
  GPIO\_PORTB\_DIR\_R && \texttt{~BIT0;}\\

  //set pin 1 to Tx or output\\
  GPIO\_PORTB\_DIR\_R |\texttt{=} BIT1;\\

  //continued on next slide

http://class.ece.iastate.edu/cpre288
//calculate baudrate
uint16_t iBRD = //use equations
uint16_t fBRD = //use equations
//turn off uart1 while we set it up
UART1_CTL_R &= ~(UART_CTL_UARTEN);
//set baud rate
UART1_IBRD_R = iBRD;
UART1_FBRD_R = fBRD;
//set frame, 8 data bits, 1 stop bit, no parity, no FIFO
UART1_LCRH_R = UART_LCRH_WLEN_8;
//use system clock as source
UART1_CC_R = UART_CC_CS_SYSCLK;
//re-enable enable RX, TX, and uart1
UART1_CTL_R = (UART_CTL_RXE | UART_CTL_TXE | UART_CTL_UARTEN);
} //END of uart_init()
// Blocking call that sends 1 char over UART 1

void uart_sendChar(char data)
{
    // wait until there is room to send data
    while(UART1_FR_R & 0x20)
    {
    }

    // send data
    UART1_DR_R = data;
}
//Blocking call to receive over uart1
//returns char with data

char uart_receive(void) {
    char data = 0;

    //wait to receive
    while(UART1_FR_R & UART_FR_RXFE) {
    }

    //mask the 4 error bits and grab only 8 data bits
    data = (char)(UART1_DR_R & 0xFF);

    return data;
}
UARTDR Warning!

- UARTDR is a 32 bit register that uses 12 bits
- 4 error bits and 8 data bits
  - OE and BE deal with FIFO operations
  - PE is Parity Error
  - FE is Framing Error

http://class.ece.iastate.edu/cpre288
UART Interrupts part 1: Initialize

// turn off uart1 while we set it up
UART1_CTL_R &= ~(UART_CTL_UARTEN);

// clear interrupt flags
UART1_ICR_R = (UART_ICR_TXIC | UART_ICR_RXIC);

// enable send and receive raw interrupts
UART1_IM_R |= UART_IM_TXIM | UART_IM_RXIM;

// set priority of usart1 interrupt to 1. group 1 bits 21-23
NVIC_PRI1_R |= 0x00200000;

// enable interrupt for IRQ 6 set bit 6
NVIC_EN0_R |= 0x00000040;

// tell cpu to use ISR handler for uart1
IntRegister(INT_UART1, UART1_Handler);

// enable global interrupts
uart_enableIRQ();
IntMasterEnable();

// re-enable enable RX, TX, and uart1
UART1_CTL_R = (UART_CTL_RXE | UART_CTL_TXE | UART_CTL_UARTEN);
/Interrupt handler for uart1

void UART1_Handler(void){

    //received a byte
    if(UART1_MIS_R & UART_MIS_RXMIS){

        //do something

        UART1_ICR_R |= UART_ICR_RXIC; //clear interrupt

    }

    //sent a byte
    else if(UART1_MIS_R & UART_MIS_TXMIS){

        //Do something

        UART1_ICR_R |= UART_ICR_TXIC; //clear interrupt

    }

}
Lab 5

• Part I. Receive and Display Text
  – Check frame format and baud rate
  – Optional: Use interrupt

• Part II. Provide Character Echo
  – Send back received characters

• Part III. Push Button Response
  – Send back special messages when a push button is pressed
  – Part II should still work

• Part IV. WiFi (115200 baud)
  – Perform UART communication on top of WiFi